Defect-Based Test: A Key Enabler for Successful Migration to Structural Test

Sanjay Sengupta, MPG Test Technology, Intel Corp.
Sandip Kundu, MPG Test Technology, Intel Corp.
Sreejit Chakravarty, MPG Test Technology, Intel Corp.
Praveen Parvathala, MPG Test Technology, Intel Corp.
Rajesh Galivanche, MPG Test Technology, Intel Corp.
George Kosonocky, MPG Test Technology, Intel Corp.
Mike Rodgers, MPG Test Technology, Intel Corp.
TM Mak, MPG Test Technology, Intel Corp.

Index words: structural test, functional test, ATE, DPM, logic test, I/O test, cache test, AC loopback test, inductive fault analysis, fault models, stuck-at fault, bridge fault, delay fault, open fault, defect-based test, ATPG, fault simulation, fault modeling, DPM, test quality, fault grading, design-for-test

Abstract

Intel's traditional microprocessor test methodology, based on manually generated functional tests that are applied at speed using functional testers, is facing serious challenges due to the rising cost of manual test generation and the increasing cost of high-speed testers. If current trends continue, the cost of testing a device could exceed the cost of manufacturing it. We therefore need to rely more on automatic test-pattern generation (ATPG) and low-cost structural testers.

The move to structural testers, the new failure mechanisms of deep sub-micron process technologies, the raw speed of devices and circuits, and the compressed time to quality requirements of products with shorter lifecycles and steeper production ramps are adding to the challenges of meeting our yield and DPM goals. To meet these challenges, we propose augmenting the structural testing paradigm with defect-based test.

This paper discusses the challenges that are forcing us to change our testing paradigm, the challenges in testing the I/O, cache and logic portions of today's microprocessors, due to the paradigm shift, and the problems to be solved to automate the entire process to the extent possible.

Introduction

Traditionally, Intel has relied on at-speed functional testing for microprocessors as this kind of test has historically provided several advantages to screen defects in a cost-effective manner. Unlike other test methods, functional testing does not require the behavior of the device under test (DUT) to be changed during the test mode. Thus, functional testing allows us to test a very large number of "actual functional paths" at speed using millions of vectors in a few milliseconds; to thoroughly test all device I/Os with "tester-per-pin" ATE technology; and to test embedded caches in a proper functional mode. In addition, the testing is done in a noise environment comparable to system operation. However, functional testing is facing an increasing number of obstacles, forcing Intel to look at alternative approaches.

We begin this paper by describing the problem of continuing with functional testing of microprocessors. We then define an alternative paradigm, which we call structural test. Finally, the challenges that we face and the problems that need to be solved to test the logic, I/O, and cache subsystems of the microprocessor to make the alternative test method work are discussed.

Structural testing has been in use in the industry for quite some time. In order to meet Intel's aggressive yield and DPM goals, we propose enhancing the structural test flow, by using defect-based test (DBT). DBT is based on generating manufacturing tests that target actual physical defects via realistic fault models. The primary motivation in augmenting structural testing with DBT is to make up for some of the potential quality losses in migration to structural test methods as well as to meet the challenges of sub-micron defect behavior on the latest high-performance microprocessor circuits. Although the impact of DBT on defects per million products shipped is not well characterized, prelimi-

nary studies of DBT [1] show that it improves quality.

DBT requires a whole suite of CAD tools for its successful application. In section 5, we discuss tool requirements for successful DBT for the latest high-performance microprocessors.

The Microprocessor Test Problem

Stated simply, the increasing cost of testing microprocessors to deliver acceptable product quality on ever faster and more complex designs is the main problem we face. The cost challenges range from the non-recurring design and product engineering investment to generate good quality tests to the capital investment for manufacturing equipment for test.

Automatic Test Equipment (ATE) Cost

Following Moore's Law for the past two decades, the silicon die cost of integrated circuits has decreased as the number of transistors per die has continued to increase. In contrast, during the same period, the cost of testing integrated circuits in high-volume manufacturing has been steadily increasing. Silicon Industry Association (SIA) forecasts, depicted in Figure 1, predict that the cost of testing transistors will actually surpass the cost of fabricating them within the next two decades [2].

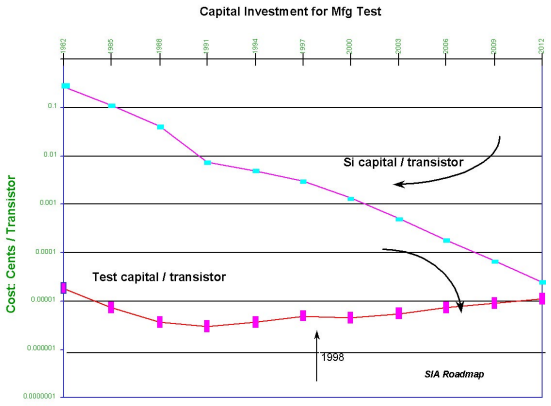

Figure 1: Fabrication and test cost trends

Lagging ATE Technology

Aggressive performance targets of Intel's chip set and microprocessor products also require increasingly higher bus bandwidth. Due to problems such as power supply regulation, temperature variation, and electrical parasitics, tester timing inaccuracies continue to rise as a function of the shrinking clock periods of high-performance designs. The graph in Figure 2 shows trends for device period, overall tester timing accuracy (OTA), and the resulting percentage yield loss. It was derived from information in the SIA roadmap.

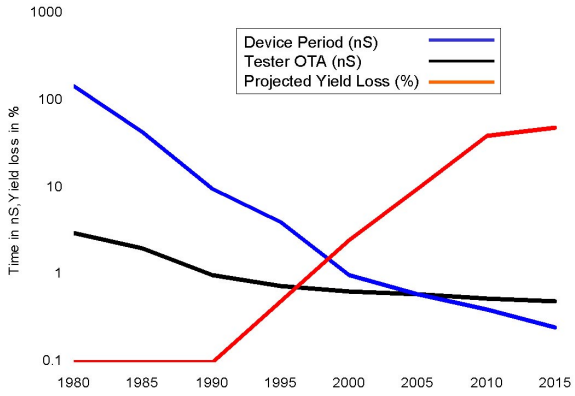

Figure 2: Tester accuracy and projected yield loss trends

In addition to the increase in device frequency and the number of I/O pins, advanced signaling techniques are also used to improve I/O performance. One such signaling innovation is the use of the source-synchronous bus, which has been in use since the Pentium® Pro line of microprocessors. Data on such a bus is sent along with a clock (strobe) generated from the driving device. This complicates testing since the ATE needs added capability to synchronize with the bus clock (strobe).

Test Generation Effort

Manual test writing, which has been in use at Intel, requires a good understanding of the structure of the DUT (typically a design block owned by a designer), as well as global knowledge of the micro-architecture. The latter is required since tests have to be fed to the DUT and the response read from the DUT. With increasing architectural complexities such as deep pipelining and speculative execution, increasing circuit design complexity and new failure modes, the cost of test writing is expected to become unacceptable if we are to meet time-to-volume targets. This is supported by the data presented in Figure 3 where manual test generation effort is compared with the effort required if ATPG, augmented with some manual test generation, were used. Note that manual test writing effort required for functional testing has been increasing exponentially over the last several generations of microprocessors at Intel. Compared to that, the projection for ATPG is very small. Note that the data for Willamette/Merced™ and beyond are projections.

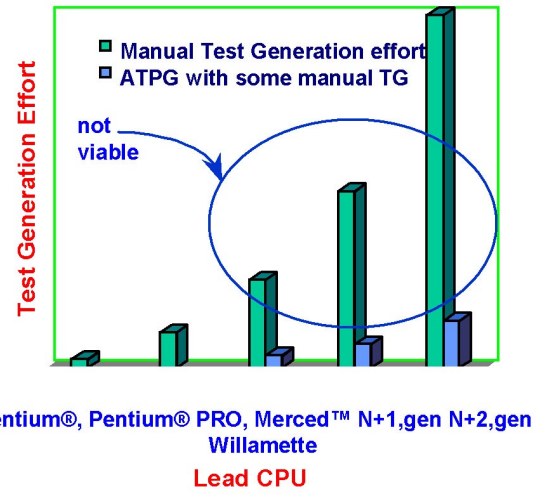

Figure 3: Test generation effort trend

Deep Sub-Micron Trends

As the feature length of transistors scales down, power supply voltage is scaled down with it, thereby reducing noise tolerance. Metal pitch is also scaled in tandem to realize the density gain. If interconnects were scaled proportionately in both pitch and height, line resistivity would rise quadratically, thereby degrading performance. To hold this trend down, metal height is scaled down by a smaller factor than the pitch, which results in increased cross capacitance.

The increase in the number of metal layers introduces more masking steps and can skew the random defect distribution towards interconnect failure modes such as bridges and open vias. Susceptibility to process variation is heightened due to the higher cross capacitance and reduced noise tolerance.

Like other CAD tools, performance validation tools are struggling to keep up with increasing design sizes and circuit design complexity. The most common solution is to build simplifying assumptions into the tools, and to offset this by the use of conservative nominal delays. Faced with increasing performance goals, designers build devices with negative timing margins. Such aggressive designs styles, coupled with increasing layout density, mean that even minor defects or process variations, which would otherwise be benign, could result in failures. Deliberate design marginality thus translates into test problems. Writing functional tests for subtle failure modes, which are made manifest under a very specific set of conditions, is becoming increasingly difficult.

Test Paradigm Shift and Challenges of the New Test Paradigm

Test paradigms are defined by (a) the *kind of test*; (b) the *kind of tester* that stores and delivers the test; and (c) the test *delivery mechanisms*.

Tests can be either *structural* or *functional*. Structural tests target manufacturing defects and attempt to ensure the manufacturing correctness of basic devices such as wires, transistors, etc. Functional tests, on the other hand, target device functionality and attempt to ensure that the device is functioning correctly. Functional tests are written primarily for architectural verification and silicon debug. They can be used for manufacturing testing also, as is done at Intel. Structural tests, on the other hand, are used primarily for manufacturing testing.

Testers come in two varieties: *functional* and *structural*. Functional testers can drive a large number of I/O pins at high clock rates with great timing accuracy. On the other hand, structural testers are limited in the number of I/O pins they can drive, as well as the speed and accuracy with which they can deliver data to the I/O pins. The cost of structural testers is considerably lower than the cost of functional testers.

Tests can be delivered in one of two ways. The device's normal functional channels are used and the device runs at operating speed. Alternatively, special design-for-test (DFT) channels can be designed, and tests are applied through these channels at less than operational speed. The scan structure and ArrayDAT exemplify this.

The test paradigm in use at Intel so far uses functional testers and functional tests. These tests are delivered using the functional channels. Functional tests are written manually. Using functional testers requires huge capital investment over short periods of time since they become obsolete very quickly. Hence, Intel is now relying more on reusable low-cost testers.

As the data showed, manual test writing for future microprocessors is not feasible. Therefore, use of ATPG tools becomes essential to meet cost and time-to-quality requirements. Thus, the paradigm that has evolved is to use low cost structural testers and use ATPG to generate the required tests. The tests being generated are structural tests. The structural tests we generate differ from the classical structural tests in that we target defects via some novel fault models. We elaborate on this later in the paper. We next discuss the challenges that this paradigm shift brings with it.

Test Generation

The loss in accessibility to the I/O pins of the device has a major impact on the ability of engineers to write functional tests for the chip. It may be possible to

load functional tests through direct access to an on-chip cache, and run them from there, but it is difficult to generate tests that operate under this mode. As a result, most of the fault-grading tests that are applied through DFT methods have to be generated using ATPG tools.

ATPG for large high-performance designs poses unique problems. Today's microprocessors have multiple clock domains, operating at different speeds. Clock gating for low-power operation is pervasive. Typical designs have many complex embedded arrays that need to be modeled for the ATPG tool. Industry standard DFT techniques, such as full scan, are often too expensive in either die area, or performance or both [7].

Defect mechanisms in deep sub-micron designs are often manifested as speed failures under very specific conditions. Most commercial ATPG tools, which are based on the stuck-at and transition fault models, are not equipped to handle these complex failure modes.

Design for Test

Although ATPG technology has progressed during this time, the success of these tools is predicated on providing a high degree of access, controllability, and observability to the internals of the design by using DFT techniques.

Scan design, the best-known structured DFT technique, comes at the cost of both performance and area, although some trade-off is possible. In order to meet tight timing requirements, high-performance designs tend to have very few gates between storage elements, which results in a high latch-to-logic ratio. Therefore, implementing scan DFT generally translates into sacrificing considerable silicon real estate.

Another DFT technique that is gaining acceptance in the industry is Built-In Self-Test (BIST), which incorporates mechanisms to generate stimuli and compress responses for later off-chip comparisons into the design. BIST allows a large number of patterns to be applied at speed in a short time, with very little tester support. However, most logic BIST techniques that enjoy commercial success today require full scan, or close to it. In addition, they need design changes to enhance random-pattern testability, to allow at-speed test application, and to prevent the system from getting into an unknown state that can corrupt the compressed response.

Such intrusive DFT techniques cannot be applied across the board to high-performance devices, so logic BIST for microprocessors has only limited applicability today. High-volume, high-performance microprocessors have to choose between the high cost of scan DFT or resort to more custom access methods of getting stimuli to, and observing responses at, the boundaries of internal components.

Test Application Methodology

Industry data shows that testing a device using functional tests rather than other test patterns results in fewer escapes [4]. A possible explanation is that when the device is exercised in functional mode, defects that are not modeled, but affect device functionality, are screened out.

ATPG patterns differ fundamentally from functional test patterns: they explicitly target faults rather than checking for them by exercising the functionality of the device, and they are typically very efficient, detecting each fault fewer times in fewer ways. Also, since they are based on using DFT structures to apply tests, they are applied at a lower speed. Consequently, there is a risk of losing "collateral" coverage of defects that do not behave like the modeled faults.

Structural testers have a small set of pins that operate at a lower frequency than the device and contact only a subset of its I/O pins. The device needs to be equipped with special DFT access ports to load and unload the vectors from the tester. The boundary scan test access port, scan input and output pins, and direct access test buses are typically for this purpose.

A few seconds of functional test may apply millions of patterns to a chip. In contrast, due to power, noise, and tester bandwidth considerations, the serial loading of test vectors from the DFT ports may be slow, and the number of test vectors that can be applied from the structural tester may be far fewer than in a functional test environment. This has implications for the quality of the structural test set.

Speed Test

Unlike many standard parts, microprocessors are binned for speed. This necessitates speed test, where the objective is to determine the maximum frequency at which the part can be operated. In the past, a small set of the worst speed paths was identified, and tests written to exercise these paths were used to characterize the speed of the device. With increasing die sizes and shrinking device geometry, in-die process variation is becoming significant. It is no longer safe to assume that all paths will be affected equally, and a larger set of representative paths needs to be tested to determine the maximum operating frequency.

One of the implications of applying vectors in the DFT mode is that the device may not be tested in its native mode of operation. Special-purpose clocking mechanisms are implemented to apply the tests to the targeted logic blocks after they have been loaded. The electrical conditions, background noise, temperature, and power supply may all be different in the DFT mode.

These factors introduce inaccuracies, necessitating guard-bands, in measuring the speed of the device.

I/O Timing Test

Traditional I/O functional testing relies on the ability of the tester to control and observe the data, timing, and levels of each pin connected to a tester channel. The testing of the I/O buffers can be divided into three basic categories: timing tests (e.g., setup and valid timings), level tests (e.g., Vil and Vol specifications), and structural tests (e.g., opens and shorts). The timing specifications of the I/O buffers are tested during the class functional testing of the device. With the use of structural testers, dedicated pin electronics are no longer available on the tester to make timing measurements on each I/O pin on the device.

Assuming that the I/O circuit meets the design target and that timing failures are results of defects at the I/O circuits, the problem of testing complex timing becomes one of screening for these defects, instead of the actual timing specification itself.

Defect-Based Test

Applicability of the Stuck-At Fault Model

Although functional patterns are graded against the single stuck-at fault model, it is well known that most real defects do not behave like stuck-at faults. Instead, stuck-at fault coverage has been used as a stopping criterion for manual test writing with the knowledge that the functional tests would catch other types of defects that impact device functionality. This measure of test quality worked quite well for a long time. However, in the recent past, there is conclusive data from sub-micron devices that proves that the outgoing DPM can be further reduced by grading and developing functional tests using additional fault models such as bridges etc. Therefore, the success of the single stuck-at fault model cannot be guaranteed as we move further into the sub-micron devices.

The quality of ATPG patterns is only as good as the quality of the targeted fault models. As the test environment forces the transformation from functional to structural testing, there is yet another strong case for the development of better test metrologies than the simplified stuck-at fault model. Defect-based test addresses this risk by using better representations of the underlying defects, and by focusing the limited structural test budget on this realistic fault.

What is Defect-Based Test?

Before we define defect-based test, we distinguish between two terms: defect and fault model. Defects are physical defects that occur during manufacturing. Examples of defects are partial or spongy via, the presence of extra material between a signal line and the V_{dd} line, etc. Fault models define the properties of the tests that will detect the faulty behavior caused by defects. For example, stuck-at 1 tests for line *a* will detect the defect caused by a bridge between the signal line *a* and V_{dd}.

It has been reported in the literature [5] that tests that detect every stuck-at fault multiple times are better at closing DPM holes than are tests that detect each fault only once. This approach, called N-detection, works because each fault is generally targeted in several different ways, increasing the probability that the conditions necessary to activate a particular defect will exist when the observation path to the fault site opens up.

Defect-based tests are derived using a more systematic approach to the problem. First, the likely failure sites are enumerated. Each likely defect is then mapped to the appropriate fault model. The resulting defect-based fault list is targeted during ATPG. Tests generated in this way are used to complement vectors generated using the stuck-at fault model. Unlike the stuck-at model that works off of the schematic database, the starting point for defect-based test is the mask layout of the device under test. Layout-based fault enumeration is a cornerstone of defect-based test.

The use of better fault models is expected to enhance any test generation scheme (ATPG, built-in self-test, or weighted random pattern generation) because it provides a better metric for defect coverage than does the stuck-at fault model.

Although not a proven technology, defect-based test is a strong contender for addressing some of the risks of migrating from functional to structural test. The DBT effort at Intel is aimed at proving the effectiveness and viability of this approach. The following sections describe the key problems that have to be solved, the specific tooling challenges in automating defect-based test, and a system architecture showing DBT modules in the overall CAD flow.

Challenges of Defect-Based Test

Enumerating Defect Sites

The number of all possible defects on a chip is astronomical, and it is neither feasible nor worthwhile to generate tests for all of them. Fault enumeration is the task of identifying the most important defect sites and then mapping them into fault models that can be targeted by fault simulation and ATPG tools.

To enumerate likely defect sites, we need to understand the underlying causes of defects. Broadly speaking, defects are caused by process variations or random localized manufacturing imperfections, both of which are explained below:

- *Process variations* such as transistor channel length variation, transistor threshold voltage variation, metal interconnect thickness variation, and inter metal layer dielectric thickness variation have a big impact on device speed characteristics. In general, the effect of process variation shows up first in the most critical paths in the design, those with maximum and minimum delays.
- *Random imperfections* such as resistive bridging defects between metal lines, resistive opens on metal lines, improper via formations, shallow trench isolation defects, etc. are yet another source of defects. Based on the parameters of the defect and "neighboring parasitic," the defect may result in a static or an at-speed failure.

Techniques used for the extraction of faults due to random defects and process variations may differ, but the fundamental approach is to identify design marginalities that are likely to turn into defects when perturbed. The output of a fault extraction tool is typically ordered by probability of occurrence.

Defect Modeling

To test a device, we apply a set of input stimuli and measure the response of the circuit at an output pin. Manufacturing defects, whether random or systematic, eventually manifest themselves as incorrect values on output pins.

Fault simulators and ATPG tools operate at the logical level for efficiency. A fault model is a logic level representation of the defect that is inserted at the defect location. The challenge of fault modeling is to strike a balance between accuracy and simplicity as explained below:

- *Accuracy*. The output response of the logic-level netlist with the fault model inserted should closely approximate the output response of the defective circuit for all input stimuli.
- *Simplicity*. The fault model should be tractable, i.e., it should not impose a severe burden on fault simulation and ATPG tools.

During the model development phase, the effectiveness of alternative models is evaluated by circuit simulation. Vectors generated on the fault model are simulated at the circuit level in the neighborhood of the defect site, using an accurate device-level model of the defect. However, due to the number of possible defect sites and the complexity of circuit simulation, this can only be done for a small sample.

Defect-Based Fault Simulation

Simulation of defect-based models is conceptually similar to stuck-at fault simulation, with a couple of twists:

- The number of possible defect-based faults is orders of magnitude larger than stuck-at faults, so the performance of the tool is highly degraded. In order to be effective, a defect-based fault simulator has to be at least an order of magnitude faster.
- Defect-based faults may involve interactions between nodes across hierarchical boundaries, making it impractical to use a hierarchical or mixed-level approach to fault simulation. It is necessary to simulate the entire design at once, which also imposes capacity and performance requirements.

Defect-Based Test of Cache Memories

Background: The Growth of Caches for Microprocessors

The use of caches for mainstream microprocessors on Intel® architectures, beginning in the early 90s with the i486™ processor, heralded a return to Intel's original technical core competency, silicon memories, albeit with several new twists. The embedded CPU caches have increased in size from the 4K byte cache of the i486 processor generation to 10s and 100s of kilobytes on today's processors and to even larger embedded CPU caches being considered for the future. This has resulted in a steady increase in the fraction of overall memory transistors per CPU and in the amount of CPU cache die area throughout the last decade.

A second key cache test challenge is the increasing number of embedded arrays within a CPU. The number of embedded memory arrays per CPU has gone from a handful on the i486 and i860™ processors to dozens on the more recent Pentium® Pro and Pentium® II processor lines.

Memory Testing Fundamentals: Beyond the Stuck-At Model

The commodity stand-alone memory industry, i.e., DRAMs and 4T SRAMs, have evolved fairly complex sets of tests to thoroughly test simple designs (compared to the complexity of a modern microprocessor) [6]. The targeted fault behaviors include stuck-at, transition, coupling, and disturbs, and the resulting number of targeted tests per circuit, per transistor, or per fault primitive on a memory is much higher than for digital logic devices. On VLSI logic, the chal-

lenge is to achieve stuck-at fault coverage in the upper 90 percentile, while on stand-alone memories, the number of targeted tests per circuit component is typically in the 100s or more likely 1000s of accesses per bit within a robust memory test program.

One reason for the greater complexity of memory tests is that at the core of a typical digital memory is a sensitive, small signal bit, bit bar, and sense amp circuit system. Even for stand-alone memories, access and testing of the analog characteristics (e.g., gain, common mode rejection ratio, etc.) is not directly possible and must be done indirectly through the digital interface of address and data control and observability. A large number of first order variables subtly affect the observability of silicon memory defect behavior. Therefore, most memory vendors characterize each variant of a given product line empirically against a broad range of memory patterns before settling on the test suite that meets quality and cost considerations for high-volume manufacturing. These characterization test suites (also known as "kitchen sink" suites) consist of numerous algorithmic march patterns and different sets of cell stability tests (e.g., data retention, bump tests, etc).

A key concept for robust memory testing is the logical to physical mapping. On a given physical design of an array, the physical adjacencies and ordering of bits, bit lines, word lines, decoder bits, etc., typically do not match the logical ordering of bits (such as an address sequence from bit 0 to bit 1 to … highest order bit. Memory tests are designed to be specifically structural where worst-case interactions of the implemented silicon structures with true physical proximity are forced. Thus the true physical to logical mapping is a subsequent transform that must be applied to a given memory pattern in order to maximize its ability to sensitize and observe defects and circuit marginality. Correct and validated documentation to the downstream test writer of the actual physical-to-logical mapping is as important as other design collateral.

Embedded Cache Testing and DFT in the Context of Logic Technologies

Testing of embedded caches also needs to consider the context of related logic technologies. To start with, the basic embedded cache memory cell is typically a six transistor (6T) SRAM as compared to the more typical DRAMs and four transistor (4T) SRAM of the stand-alone silicon memory industry. The 6T SRAM offers better robustness against soft errors and can be thoroughly tested to acceptable quality levels with somewhat simpler test suites. However, the critical motivating factor is that a 6T SRAM cell is feasible, within the context of a high-performance logic silicon fabrication process technology, without additional process steps.

The smaller size (area, # bits) and 6T cell of the embedded CPU cache make it less sensitive than the stand-alone commodity 4T SRAMs and DRAMs. This is somewhat offset by the fact that embedded caches are generally pushing the SRAM design window on a given fabrication technology for system performance reasons. Therefore, adequate testing of embedded 6T SRAMs requires an optimal use of robust memory test techniques targeted at defect behaviors, such as complex march algorithms and cell stability tests.

A critical challenge for embedded SRAM caches is the architectural complexity of access and observability of such arrays compared to a stand-alone memory. For example, for an embedded array such as an instruction cache or a translation buffer, there may not be a normal functional datapath from the array output to the chip primary outputs, making writing of even the simplest memory algorithmic patterns such as the 10N March C- an extreme challenge for even the most experienced CPU design engineers and architects.

In the end, the number and variety of caches and embedded arrays in today's microprocessors demand a multiple of DFT and test solutions optimized to the physical size and area of the various arrays, the performance and cost boundary conditions, and the architectural and micro-architectural details of an embedded array's surroundings. Circuit-level DFT, such as WWTM [7], can offer targeted structural coverage, in this case against cell stability issues and weak bits. External access via special test modes or self-test (BIST) circuits may provide the better solution within different sets of variables. However, care must be taken to ensure the completeness and correctness of the solution in any case and that some level of structural approach is used, i.e., appropriate stimulus-response mapped to the physical implementation of the memory structures. Different types of memory structures, e.g., small signal SRAMs, full Vcc rail swing CMOS register files, CAMs, or domino arrays, each require a targeted structural approach mapped to their strength and weaknesses with respect to defect response.

Technology Development Strategy

The technology for defect-based test spans multiple disciplines in design, CAD tooling, and manufacturing. Although individual components have been tried both within Intel as well in academia and industry, real data on high-volume, high-performance microprocessors is needed to establish the value of this approach.

The defect-based test program at Intel emphasizes early data collection on the effectiveness of fault models. Partnerships with design teams interested in pioneering these new capabilities as they are developed

form a cornerstone of this effort. Technology development proceeds in phases as follows:

- *Fault model development*. There are a large number of possible defect types that can be modeled. Defects are chosen for modeling based on frequency of occurrence, ease of modeling, escape rate, and perceived importance to the partner design team. Bridges and path delay faults will be the first set of fault models to be investigated.
- *Tool development*. A minimal set of prototype tools is developed for the enumeration and simulation of the target fault models. These tools are targeted for limited deployment to a select group of experts in the project design team. The focus of tool development is on accuracy, not performance. Where possible, the tools are validated against existing "golden" capabilities.

  Tools for defect enumeration need to leverage physical design and performance verification tools. Close co-operation with tool builders and project design automation teams is required to build on existing tools, flow, and data in order to facilitate the defect-extraction process.
- *Enumerating fault sites*. The actual task of enumerating fault sites is performed jointly by the technology development team and the design team. Working together, test holes such as new architectural enhancements or modules for which legacy tests could not be effectively ported are identified. Fault grading resources are allocated for defect-based test on those regions. When available, data from the FABs are used to assign probabilities to defect sizes.
- *Test generation*. Defect-based tests are generated by first grading functional validation and traditional fault grade vectors, and then by targeting the undetected faults for manual test writing. Test writing is necessary at this time because a defect-based ATPG is not yet available, and the legacy designs on which the technology is being pioneered do not have adequate levels of structured DFT. To contain the cost of test writing, defect-based tests are written for carefully selected modules of the design.
- *Model validation.* Model validation requires close partnering with the product engineering team. Some changes are required to the manufacturing flow to collect data on the unique DPM contribution of the defect-based tests.

  Data from the model validation phase is fed back into model development, as illustrated below. Once a particular fault model is validated, we will enter into development (or co-development with a tools' vendor) of an ATPG capability for that model.

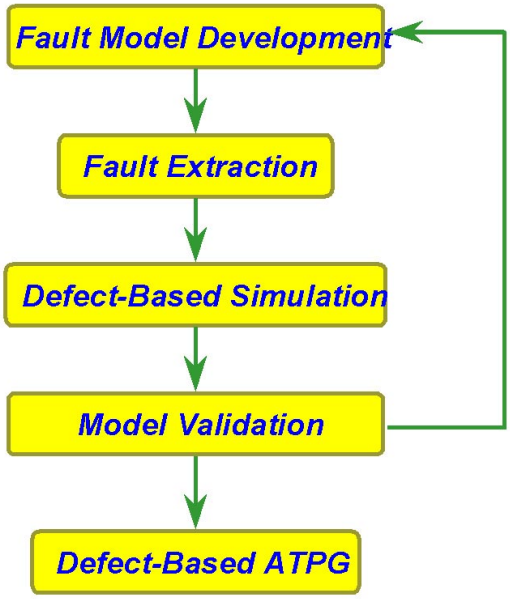

Figure 4: Technology development process flow

Defect Modeling

Modeling of Random Defects

The challenge in fault modeling is to capture a general cause and effect relationship that can be easily simulated or targeted in the case of automatic test pattern generation. A degenerate case of this general approach is a line stuck-at fault model where output at a node is always a logical zero or always a logical one regardless of the logic value it is driven by. Another popular fault model that has been used to target random speed failures is the transition fault model, which is essentially a stuck-at fault with the addition of the condition that the faulty node make a transition, i.e., be at the opposite logic value in the cycle prior to detection.

In creating a realistic fault model for a defect, we must avoid explicitly tabulating the behavior of the defect for every state of the circuit. A table-driven approach will not lend itself to a scalable automated solution for design sizes that exceed 5 million primitives. The approach we use here is to transcribe the deviation in analog behavior into simple conditional logical deviations.

There are a large number of possible failure mechanisms that cause random defects. Rather than de-

velop models for them all and then launch into model validation, our approach is to stage the development of models and tools to address defect types in the order of their importance, and to intercept designs with a complete prototype flow for each model as they become available. This allows us to collect data on the DPM impact of defect-based test early on, and it provides feedback that we can use to refine our models.

One of the most common defect types today is interconnect bridges. As metal densities increase, the importance of metal bridges as a defect-inducing mechanism will grow. Interconnect bridging defects exhibit a range of behavior based on different values of bridge resistance.

This effect is illustrated for the circuit in Figure 5. There is a bridge defect between node *j* and *k* in this example. Node *k* is held at logic 0 as *j* changes from 0 to 1. The signal transition is propagated and observed at output *v*.

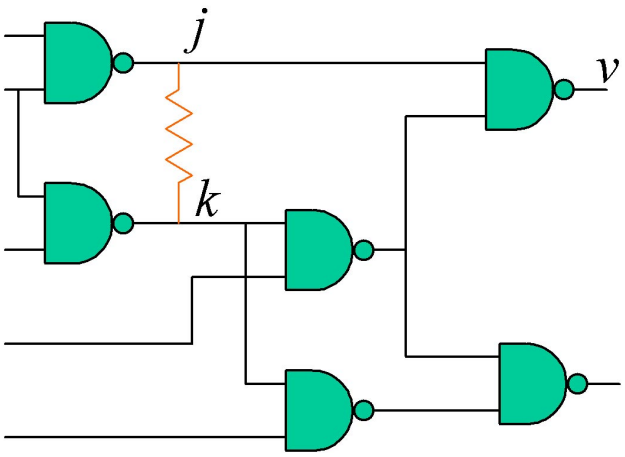

Figure 5: Example circuit with a bridge defect

Figure 6 shows the output response of the circuit for different values of the bridge resistance. Threshold voltages are marked using horizontal dashed lines, and the vertical dashed line shows the required arrival time at node v for the transition to be captured in a downstream latch.

The plot shows three distinct circuit behaviors for varying bridge resistance. For low resistance values, the output never reaches the correct logic value, and the defect shows up as a static logic failure. For intermediate resistances, the output goes to logic 0 too late, resulting in a speed failure. Very high bridge resistances are benign from the viewpoint of correct logical operation of the circuit.

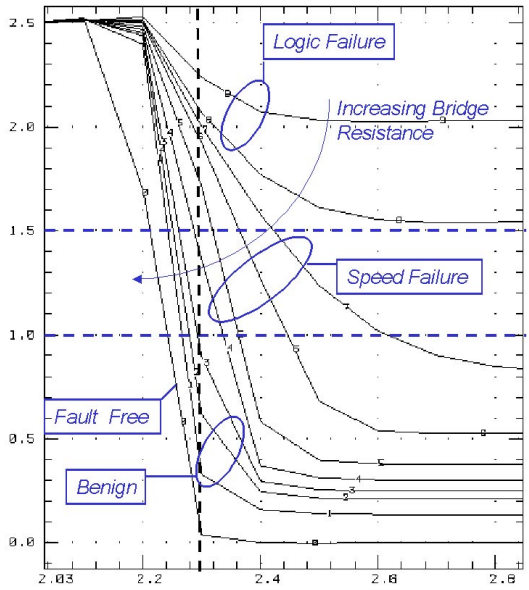

Figure 6: Output responses for a range of bridge resistance

For low resistances, the defect can be modeled as node *j* stuck at logic 0 with the condition that *k* is at logic 0. Speed failures can be modeled as a slow-to-rise transition at node *j*, with the condition that *k* is held at logic 0. Such fault models, based on generalizations of the conventional stuck-at and transition faults, are called constrained fault models.

As feature sizes are scaled down, the metal pitch is reduced in tandem to increase density. Reduced metal pitch in turn imposes limitations on the height of metal interconnects that must also decrease to improve manufacturability. Thus the line resistance per unit length goes up almost quadratically. Sustained yield requirement dictates that defect densities remain the same, which in turn implies that interconnect bridge defects also scale in dimension. Higher bridge resistance coupled with lower device resistance during ON state results in more speed failures than hard failures as illustrated in Figure 6 above.

Modeling of Systematic Defects

Not all defects are of a random nature. Known factors such as reticule position, die location on a wafer, mask imperfections, polysilicon density, device orientation, etc., cause systematic variation across wafers and dice. These effects are expected to gain prominence due to reduced noise tolerance as well as a general increase in systematic variability because of such factors as migration towards 300mm wafers, lithographic equipment, and material re-use.

Steeper production ramps are putting increasing pressure on cutting down the time for design correction and test creation based on silicon data. Thus modeling such effects is critical to the success of test.

Process variations lead to delay problems. Therefore, using information of process variations in speed test target selection needs to be addressed.

Defect-Based Test Tooling Challenges

Defect Enumeration

The goal of defect enumeration is to prune the list of all possible defects to a manageable number of the most likely faults. Because the likelihood of a fault has a strong dependence on layout geometry, process parameters and timing marginality, defect enumeration is a multi-disciplinary problem.

Here we describe layout-driven and timing-driven approaches to fault enumeration, and we discuss the inherent challenges.

Physical Design Inductive Fault Analysis

Inductive Fault Analysis (IFA) is based on the premise that the probability of a defect occurring at a particular site is a function of the local layout geometry and the distribution of failure mechanisms observed for the manufacturing process. The most commonly observed defects can be classified into two broad categories of physical faults:

- Bridges occur when the defect causes a conducting path between two nodes that are electrically isolated by design. The resistance of the bridge can vary by process, layer, and defect mechanism.
- Breaks happen when the defect introduces undesired impedance along a conducting path. In an extreme case, a break can result in an open circuit.

These physical fault models are then mapped onto logical fault models that can be used for fault simulation at the logical, or gate level, of abstraction. If the likelihood of the defect mechanism causing opens and breaks is known for the process, the physical fault sites extracted by IFA are weighted by probability. These probabilities can be used for pruning the fault list, and for expressing the fault coverage obtained by fault simulation in terms of the overall probability of catching a defective part. This weighted fault coverage number can be a better predictor for outgoing DPM than stuck-at fault coverage.

Traditionally, IFA has focussed on layout geometry and defect distribution, and it has ignored the testability of a fault. This last parameter is an important one: If the faults identified using IFA are highly testable, i.e., easily covered by tests for stuck-at faults, then using an IFA-based approach will not yield a significant incremental DPM improvement over a standard stuck-at fault model. Examples of highly likely and highly testable faults are bridges to power rails and clock lines. Therefore, the challenge for effective IFA tools is to identify faults that are both highly likely and relatively difficult to detect using stuck-at fault vectors.

Because they work at such a low level of abstraction, IFA tools need to be scalable in order to be effective on increasingly larger designs. Two divide-and-conquer approaches can be applied to the problem:

- *Hierarchical analysis*. This is where layout blocks are analyzed at a detailed level for bridges and breaks on cell-level nodes, and at a global level to analyze inter-block connectivity. The obvious drawbacks of this method are that interactions between wires across blocks, and between block-level and chip-level layout, are ignored. This problem is accentuated by the increasing trend toward over-the-cell global routing.
- *Layout carving, or "cookie-cutting."* In this approach, the layout is flattened and carved into manageable pieces called "cookies." Each cookie includes the layout to be analyzed, as well as sufficient surrounding context. A second phase is required to roll up the results collected at the cookie level, and to tie up the inter-cookie interactions.

Timing-Driven Analysis

As mentioned in a previous section, the performance verification tools for large microprocessor designs are not entirely fool proof. To begin with, the PV database is made up of data from different sources, some of which are SPICE-like simulations (very accurate) and some of which are simple estimators. The net result of this could be incorrectly ordered critical paths (speed-limiting circuit paths). During silicon debug and characterization, some of these issues are generally uncovered.

However, some serious issues abound as we look into the future. First, the increased on-die variation in deep sub-micron technologies means that different paths on the chip can be impacted differently. Further, the trend towards higher frequencies implies fewer gates between sequential elements, which may lead to a larger proportion of the chip's paths having small margins. These two factors combined pose one of the biggest test challenges, namely, speed test.

It is no longer just sufficient to have a few *most critical paths* in the circuit characterized during silicon debug. What is required is an automatic way to enumerate all such paths and then grade the structural tests for "path delay fault" coverage. There are two main issues that need to be solved. First, PV tool limita-

tions need to be worked around (issues related to generating an ordered list of critical paths), and second, modeling issues related to mapping of paths from transistor level to gate level need to be resolved. (Fault simulation happens at the gate level.)

It is likely that this huge path list can be pruned to a more manageable size. Paths could be selected based on their criticality of speed to the design and on their diversity in composition in terms of distribution of delay amongst various constituent factors such as delays on all interconnect layers and actual devices.

Comprehensive Defect Enumeration

While layout analysis may identify potential bridge defect sites, a resistive bridge may not always manifest itself as a logic error. An example of such a situation would be if the defect site has adequate slack designed into it, an increase in delay up to the slack amount will not be ordinarily detectable. Slack may change with a change in cycle time or a change in power supply voltage, thus altering the test realities.

It is therefore required that the defect enumeration scheme be coupled with timing analysis tools, which in turn should be designed to understand the effect of the test environment (temperature, voltage, cycle time) on slack.

Defect-Based Simulation and ATPG

Traditional test automation tools need to be rethought in the context of defect-based test. The fundamental reason for the effectiveness of the stuck-at fault model is that it opens up an observation path starting from the fault site. Unfortunately, the conditions needed to cause the erroneous circuit behavior may not be created at the time the observation path is set up.

Data reported in the literature show that the effectiveness of a test set could be improved by including vectors that detect the same stuck-at fault multiple times, in different ways. This approach, called N-detection, is a random way to set up the conditions needed to activate different failure modes. Defect-based fault models take this notion a step further by specifying the actual excitation conditions, called *constraints*.

- *Excitation conditions.* These are a relatively straightforward extension to commonly used fault models. Constrained stuck-at and constrained transition faults behave like their traditional counterparts except that the fault effect becomes manifest only when an externally specified condition is met.

  Existing fault simulation and test-generation tools can be used to simulate these models by augmenting the target netlist to detect the excitation condition and to inject the fault when it occurs. However, this can be expensive in terms of netlist size for big designs. Also, depending on the location of the set of nodes involved in the constraints and the fault location, the augmenting circuitry can cause design-rule violations such as phase coloring.

- *Propagation conditions*. Certain types of physical faults (such as highly resistive bridges and opens) can manifest themselves as localized delay defects. However, the size of the delay is not always large enough to allow it to be treated as a transition, or gross delay. In such cases, the effectiveness of the test can be increased, propagating the fault effect along the paths with the lowest slack. This method implies a tie-in to the timing analysis sub-system.

- *Path delay fault simulation*. Several path delay fault models have been proposed in the literature with a view to identifying tests that are robust (less susceptible to off-path circuit delays), and to simplifying the model to ease fault simulation and test generation. Any of these fault models can be used, but there are two new considerations:

  Paths in high-performance designs are not always limited to a single combinational logic block between two sequential elements. A path can span multiple clock phases, crossing sequential elements when they are transparent. A practical path delay fault model should therefore be applicable to multicycle paths. Note that such paths may feed back onto themselves (either to the source of the path or to an off-path input).

  The second consideration is that fault simulation and ATPG are typically performed at the gate level, whereas paths are described at the switch level. When a switch-level path is mapped to the gate level, a path may become incompletely specified. There may be multiple ways to test the same gate-level path not all of which exercise the same switch-level path. This problem can be addressed by specifying gate-level conditions that will exercise the switch-level path in a manner analogous to specifying excitation conditions for random defects.

- *Circuit design styles*. High-performance designs have core engines running at very high speeds and external interfaces running at lower speeds. In addition, there may be internal subsystems that run at a different clock frequency. Test generation and fault simulation tools have to be designed to accommodate multiple clock domains running at different frequencies. The clocks are typically generated internally and synchronized. DFT design rules, particularly those that check the clocking methodology, need to be enhanced to handle such designs.

Another important design consideration is power delivery and consumption. In order to reduce a chip's power needs, clocks are often gated to dynamically turn off units that are not being used at a particular time. In the past, many tool designers assumed that clock-gating logic could be controlled directly by external pins, or they treated clock-gating logic as untestable. These assumptions are no longer valid.

- *Capacity and performance*. Next-generation CPUs are expected to require 5 to 10 million primitives to model at the gate level. The designs contain on the order of a hundred embedded memory arrays. These arrays have multiple read/write ports, with some ports accessing only parts of the address or data spaces of the array. In the past, most ATPG tools have provided support for simple RAM/ROM primitives that can be combined to model more complex arrays. However, from the point of view of database size and test generation complexity, it is essential to directly support more general behavioral models.

  Defect-based fault models impose additional performance requirements on the tools because of the exploding number of faults that need to be targeted. In order to deal with larger designs, shrinking time-to-quality goals, and the larger number of faults, the performance of test automation tools needs to increase by an order of magnitude.

Failure Diagnosis

Automated failure diagnosis is valuable at different stages of a product's life: silicon debug and qualification manufacturing test and analysis of customer returns. Next-generation failure analysis tools have two major requirements:

- They must support defect-based models. Diagnostic tools need to leverage the defect resolution provided by the new fault models. This will enhance diagnostic resolutions by narrowing down the probable cause of a failing device to one defect-based fault, where partial matches were found, before using the stuck-at fault model. Diagnostic resolution can be further enhanced by the use of defect probability for prioritizing candidate failures.
- They must support limited sequentiality for high-performance designs that cannot afford scan DFT in pipelined stages.

Defect-Based Tooling Framework

The design flow in Figure 7 shows the new CAD modules introduced for DBT and their relationship to existing design and test automation modules. The new modules are highlighted in yellow.

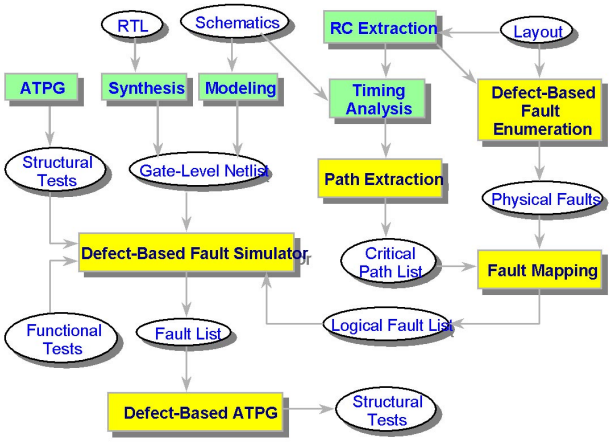

Figure 7: Defect-based test system architecture

The left half of the flow is analogous to the traditional fault simulation and ATPG flow. These tools work on a gate-level model, which is generated either top-down by synthesis of RTL, or bottom-up by logic modeling of device-level circuits. The defect-based fault simulator accepts fault lists of realistic defect models. Traditional ATPG vectors, as well as existing functional tests, are fault simulated to filter out defect-based faults that are detected by these tests. A defect-based ATPG is used to generate tests for undetected faults.

The right half of the flow is for layout and timing-driven fault enumeration, and it is new to DBT. The analogous step for traditional ATPG is stuck-at fault enumeration and collapsing based purely on gate-level analysis. Random faults are typically enumerated from the layout with the possible use of interconnect capacitances obtained by RC extraction tools. Critical paths for speed test are extracted from timing analysis. The identified fault sites exist at the layout or device level, and they need to be mapped to the logical level for fault simulation and ATPG.

Conclusion

In this paper we described the challenges faced by Intel in continuing with functional test as the primary mechanism for screening manufacturing defects, and we examined structural test as an alternative. Three major test quality risks were identified in migrating to structural test:

- Reduced test data volume due to the inefficiencies in loading test patterns from a structural tester.

- The loss of collateral defect coverage provided by functional tests that are applied at speed in the normal functional mode of operation.
- Sub-micron trends indicating that interconnect defects such as bridges and opens will dominate the defect distribution. Simulation results were presented that indicate that an increasing number of defects will result in speed failures rather than hard failures, requiring alternate ways of generating test patterns.

Defect-based test was introduced as an approach to mitigate some of these risks by increasing the effectiveness of ATPG-generated vectors. While this approach is intuitively appealing, it poses formidable challenges. Little hard evidence is available on the effectiveness of such an approach. Fault models that represent defect behavior well, and are tractable from a test generation viewpoint, have to be developed. Tools for the enumeration of likely fault sites and for test generation tools with the new fault models need to be implemented.

The technology development strategy for DBT was presented as an evolutionary cycle that builds on prototype capabilities and uses strategic partnerships with design teams. Silicon data collected from these experiments are used to refine and validate fault models and the tooling collateral as they are developed.

The tooling challenges for defect-based test for large, high-performance designs were discussed. Commercial capabilities that exist today are either insufficient, or cannot be scaled to meet the needs of next-generation microprocessor designs. These challenges span the design flow from logical to physical design, and they will require a concerted effort by the CAD industry to make defect-based test a robust, scalable solution.

Acknowledgments

The defect-based test effort spans three departments in MPG and TMG. We thank Will Howell and Paul Ryan of TT Q&R, Sujit Zachariah, Carl Roth, Chandra Tirumurti, Kailas Maneparambil, Rich McLaughlin, and Puneet Singh of Test Technology, and Mike Tripp, Cheryl Prunty, and Ann Meixner of STTD for their ongoing contributions.

We have benefited greatly from feedback received over the course of several technology reviews held with the Willamette DFT team, in particular Adrian Carbine, Derek Feltham, and Praveen Vishakantaiah.

References

[1] Schnarch, Baruch, "PP/MT Scoreboarding: Turning the Low DPM Myth to Facts," *Proceedings of the 1998 Intel Design and Test Technology Conference*, pp. 13-18, Portland, OR, July 21-24, 1998.

[2] Wayne Needham, internal memo based on data extracted from 1997 SIA Roadmap.

[3] Design and Test Chapter, *National Technology Road Map*, 1997, available on the Web (URL: www.sematech.org).

[4] "Sematech Test Method Evaluation Data Summary Report, " Sematech Project S-121, version 1.4.1, 1/30/96.

[5] S.C. Ma, P. Franco, and E.J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results," *Proceedings 1995 Int. Test Conference*, pp. 663-672, Washington, D.C., Oct. 23-25, 1995.

[6] A. J. van deGoor, *Testing Semiconductor Memories, Theory and Practice,* John Wiley and Sons, Ltd, England, 1991.

[7] A. Meixner and J. Banik, "Weak Write Test Mode: An SRAM Cell Stability Design for Test Technique," *Proc. 1996 Int. Test Conf.*, pp. 309-318.

[8] A. Carbine and D. Feltham, "Pentium® Pro Processor Design for Test and Debug," *IEEE International Test Conference*, 1997, pp. 294-303.

Authors' Biographies

Sanjay Sengupta received an MSEE in electrical and computer engineering from the University of Iowa and a B.E. (Hons.) in electrical and electronics engineering from BITS, Pilani, India. He works on the development of test tools and methodology for next-generation microprocessors, and currently manages the defect-based logic test effort. Prior to joining Intel, he worked on test automation at Sunrise Test Systems and LSI Logic. His e-mail is sanjay.sengupta@intel.com.

Sandip Kundu has a B.Tech (Hons.) degree in electronics and electrical communication engineering from IIT, Kharagpur and a PhD in computer engineering from the University of Iowa. Prior to joining Intel, he worked at IBM T. J. Watson Research Center and at IBM Austin Research Laboratory. Sandip has published over forty technical papers and has participated in program committees of several CAD conferences including DAC, ICCAD, and ICCD.
His e-mail is sandip.kundu@intel.com.

Sreejit Chakravarty received his BE in electrical and electronics engineering from BITS, Pilani and a PhD in computer science from the State University of New York. He joined Intel in 1997. Prior to that, from 1986-97, he was an Associate Professor of Computer Science at the State University of New York at Buffalo. Sreejit has published over 70 technical papers and has served on the program committee of several international conferences, including VTS and VLSI Design. He has co-authored a book on I testing. His e-mail is sreejit.chakravarty@intel.com.PPQ

Praveen Parvathala received an MSEE from New Jersey Institute of Technology in 1989. Since then he has been working at Intel. As a designer, he contributed to the development of the i860™, Pentium® and Merced™ microprocessors. Since 1995 he has been working on DFT and is currently involved in the development of defect-based DFT methods and tools in MPG's Test Technology group.
His e-mail is praveen.k.parvathala@intel.com.

Rajesh Galivanche received his MSEE from the University of Iowa in 1986. Since then he has worked in the areas of Design for Test, ATPG, and simulation. Previously, Rajesh worked at Motorola, LSI logic, and Sunrise Test Systems. Currently, he manages the Logic Test Technology development team in the Microprocessor Product Group.
His e-mail is rajesh.galivanche@intel.com.

George Kosonocky received his BSEE from Rutgers University. He presently manages MPG's Test Technology organization and has held various management positions at Intel since 1983 in design engineering, marketing, quality and reliability, and program management. George holds a patent in non-volatile memory design. His e-mail is george.a.kosonocky@intel.com.

Mike Rodgers received a BSEE from the University of Illinois in 1986. He has been with Intel for 12 years in various capacities in Microprocessor Q&R including managing A4/T11 Q&R and FA groups in Santa Clara and IJKK. He currently is responsible for TT's Cache Technology and co-chairs the TMG-MPG Test Technology Planning Committee.
His e-mail is michael.j.rodgers@intel.com.

TM Mak is working on circuit and layout-related DFT projects. He has worked in product engineering, design automation, mobile PC deployment, and design methodology for various products and groups since 1984. He graduated from Hong Kong Polytechnic in 1979. His e-mail is t.m.mak@intel.com.